Kernel methods

Kernels are an extension of the basis functions to further increase the applicability of linear methods: so far objects were represented a fixed-length feature vectors $x \in \mathbb{R}^n$ or $\phi(x)$; what about objects with variable length or infinite dimensions? (e.g. strings, image features, time series, ...)

Instead of representing each particular instance of the input space in a feature space, we will define a Kernel function.

Kernel function: a real valued function $K(x, x') \in \mathbb{R}$ that measures how x and x' are similar
$K(x, x')$ is like a distance function, so $K(x, x') \geq 0$

Just like any distance function, K must satisfy these two conditions:

- K must be symmetric: $K(x, x') = K(x', x)$
- K must be non-negative: $K(x, x') \geq 0$

These constraint are not strictly required

/ the important aspect is that K returns values close to 0 when the two values are similar

Linear classification and regression methods can be extended with this concept and we won't need to represent x anymore, but just give a function.

If the input space X is the set of all the possible sequence of words of variable length we could provide just a kernel function that given two sequence of words of different length will return a real value that is a similarity between the two.

One possibility is to define a kernel in terms of a feature space

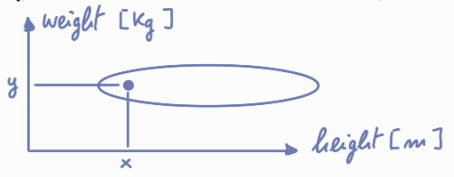
$$K(x, x') = \phi(x)^T \phi(x')$$

The notion of kernel is more general with respect to the notion of transformation of the input.

Data normalization

Data normalization is crucial when dealing with kernels and more in general with non-linear models in the input (still linear in the weights).
Input data in the dataset D must be normalized in order for the kernel to be a good similarity measure in practice.

e.g. consider a dataset reporting the weight of a group of people in relation with their height [m]. If we transform the input in such a way that the height is expressed in [cm], the dataset will stretch, nothing will change from the point of view of the representation of the problem but we will have a different solution



the dataset will stretch, we are multiplying each value by 100.

When we use similarity/distance measures we have to normalize the data

We have several techniques:

- min-max: $\bar{x} = (x - min)/(max - min)$
- unit vector: $\bar{x} = x/|x|$
- normalization (standardization): $\bar{x} = (x - \mu)/\sigma$

μ = mean
σ = standard deviation

after this normalization the data will have $\mu = 0$ and $\sigma = 1$

Kernel families

There exist different kernel families:

- Linear: $k(x, x') = x^T x'$
- Polynomial: $k(x, x') = (\beta x^T x' + \gamma)^d$, $d \in \{2, 3, \dots\}$
- Radial Basis Function (RBF): $k(x, x') = e^{-\beta |x - x'|^2}$
- Sigmoid: $k(x, x') = \tanh(\beta x^T x' + \gamma)$

These are general purpose kernels; we can define application specific kernels.

How do we use these kernels?

Consider a linear model $y(x; w) = w^T x$ with dataset $D = \{(x_n, t_n)_{n=1}^N\}$.
We define an error function $J(w)$ e.g. use the squared error function + regularization term
$J(w) = (t - Xw)^T (t - Xw) + \lambda |w|^2$
Minimize the error function $J(w)$ to get the solution (trust me on the computations)

$$\hat{w} = (X^T X + \lambda I_N)^{-1} X^T t = X^T (XX^T + \lambda I_N)^{-1} t$$

(I_N: identity matrix of dimension $N \times N$) } solution

$$\alpha = (XX^T + \lambda I_N)^{-1} t$$

then:

$$\hat{w} = X^T \alpha$$

$$y(x; \hat{w}) = \hat{w}^T x = \sum_{n=1}^N \alpha_n x_n^T x$$

(x is used to refer to a general sample while x_n is the current sample considered in the summation → x is a free variable, any possible sample e.g. new sample to classify)

If we consider a linear kernel $\boxed{k(x, x') = x^T x'}$, we can rewrite the model as:

$$y(x; \hat{w}) = \sum_{n=1}^N \alpha_n k(x_n, x)$$

(with this definition we rewrote the linear model using a Kernel)

with $\alpha = (\boxed{XX^T} + \lambda I_N)^{-1} t$

$XX^T = K$ is called Gram matrix

Before → After → Generalized Kernel

$y(x; \hat{w}) = w^T x$ → $y(x; \hat{w}) = \sum_{n=1}^N \alpha_n x_n^T x$ → $\boxed{y(x; \hat{w}) = \sum_{n=1}^N \alpha_n k(x_n, x)}$

linear model — SVM — Kernelized SVM

Solution:

$$\alpha = (K + \lambda I_N)^{-1} t$$

K:

$$XX^T = \begin{bmatrix} x_1^T x_1 & \cdots & x_1^T x_N \\ \vdots & \ddots & \\ x_N^T x_1 & \cdots & x_N^T x_N \end{bmatrix}$$

(cross product of all the possible pairs in the dataset)

$$\boxed{K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}}$$

(each component will be $k(x_i, x_j)$)

generalized solution with Kernel function

We can rewrite the problem and the solution with K.

We can use this model with every Kernel (Kernel trick or Kernel substitution).
If our model is a linear combination of the Kernel function (linear combination of the similarity between x - free variable - and all the samples in the dataset) then the solution of this model (coefficient α that minimizes the error function) is

$$\alpha = (K + \lambda I_N)^{-1} t$$

If input vector x appears in an algorithm only in the form of an inner product $x^T (x')$, replace the inner product with some Kernel $K(x, x')$ — with the samples in the dataset

- can be applied to any x (even infinite size)
- no need to know $\phi(x)$
- directly extend many well-known algorithms

Kernel trick

We explored a variety of learning algorithms based on non linear Kernels. One of the significant limitations of many such algorithms is that the Kernel function $K(x_n, x_m)$ must be evaluated for all possible pairs x_n and x_m of training points, which can be computationally infeasible during training and can lead to excessive computation times when making predictions for new data points.
Now we examine the Kernel-based algorithm that have sparse solutions, so that predictions for new inputs only depends on the Kernel function evaluated on a subset of the training data points.

Kernelized methods that use a small subset of training data points

- Kernelized SVM only uses the support vector
  - → classification
  - → regression
- Kernelized linear regression